ISSN 2007-9737

A Formula Embedding Approach to Math Information Retrieval

Amarnath Pathak[1], Partha Pakray[2], Alexander Gelbukh[3]

[1] National Institute of Technology Mizoram,
Department of Computer Science and Engineering,
India

[2] National Institute of Technology Silchar,
Department of Computer Science and Engineering,
India

[3] Instituto Politécnico Nacional, Center for Computing Research (CIC),
Mexico

{amar4gate, parthapakray}@gmail.com, gelbukh@gelbukh.com

Abstract. Intricate math formulae, which majorly constitute the content of scientific documents, add to the complexity of scientific document retrieval. Although modifications in conventional indexing and search mechanisms have eased the complexity and exhibited notable performance, the formula embedding approach to scientific document retrieval sounds equally appealing and promising. Formula Embedding Module of the proposed system uses a Bit Position Information Table to transform math formulae, contained inside scientific documents, into binary formulae vectors. Each set bit of a formula vector designates presence of a specific mathematical entity. Mathematical user query is transformed into query vector, in similar fashion, and the corresponding relevant documents are retrieved. Relevance of a search result is characterized by extent of similarity between the indexed formula vector and the query vector. Promising performance, under moderately constrained situation, substantiates competence of the proposed approach.

Keywords. Math information retrieval, formula embedding, math formula search, scientific document retrieval, precision.

1 Introduction

Information Retrieval, an application area of Natural Language Processing (NLP), intends to content information needs of the end users. Information needs vary from individual to individual, which poses challenge to the design of search engines. Though the conventional search engines come equipped with text and multimedia content searching abilities, they often lack the interface and the requisite competence to search math formulae present inside scientific documents.

Effective retrieval of the scientific documents is challenged by abundance and complexity of the math formulae contained therein. Presence of these formulae significantly demarcates scientific documents from normal text documents, and mandates modification in conventional retrieval mechanisms to ease their retrieval. Intelligence of scientific documents search engine is manifested in its ability to comprehend crux of math formula query prior to searching. Relevant search results should be substantially influenced and guided by the semantics of query formula rather than its syntactic constructs.

For example, given query formula to be $\frac{1}{1+e^x}$, usually, the end user will be equally content with search results containing $\frac{1}{1+e^p}$, $\frac{1}{1+a^x}$, $\frac{1}{1+a^p}$, $1+e^x$, $\frac{1}{1+e^{-x}}$ e^x, a^p and so on. Thus, the preciseness of numbers, variables and syntactic constructs of query formula are often compromised for the sake of semantic equivalence or sub-formulae results.

Hence, unlike text search, relevance of search results in math formula search encompasses a wide range of meanings.

Distinctive stipulations, laid out by math formula search, necessitate revamping conventional techniques to successfully meet the expectations of end users of math search engine. To this end, document indexing and formulae search techniques have been subjected to numerous modifications, which exhibit promising performance under normal as well as eccentric situations. In particular, indexing of canonicalized, tokenized and structurally unified variants of raw formulae have been found to engender relevant search results [16, 13, 14].

Canonicalization tackles representational non-uniformity of math formulae whereby equivalent formulae, with minor representational difference, are homogenized. In addition, tokenization and structural unification help retrieve sub-formulae and similar formulae, respectively. Weights assigned to tokenized and structurally unified formulae characterize extent of their similarity with original formulae, which eventually help in ranking the retrieved documents. Although the approach splendidly caters to the need of math information seekers, it can be fascinating to prospect other competent techniques, which may depict similar traits with relatively lesser effort.

In this paper, we present our formulae embedding approach to math information retrieval, and a comprehensive analysis of the obtained system results to infer strengths and weaknesses of the proposed approach. Formulae embedding envisions math formulae inside documents and the mathematical user query as binary vectors, wherein each bit position designates absence or presence of a specific mathematical entity (such as single character variable, superscript, subscript, fraction, special symbol, and so on).

After having preprocessed the scientific documents and the math formulae contained therein, each formula is transformed into a fairly large-sized vector, which encompasses nearly all the information content of the corresponding formula.

Eventually, the multitude of formula vector to corresponding document mapping are indexed to assist the searcher module in its hunt for relevant documents. Count of matching set bits (set bits at same positions) of indexed formula vector and query vector constitutes the crucial criterion for assessing relevance of the retrieved documents.

Although there are some pitfalls associated with the proposed approach, it predominantly lives up to the expectations of the end users as a baseline approach.

Efficiency of the proposed approach is tested using a corpus containing 212 documents of the NII Testbeds and Community for Information Access Research (NTCIR)-12 MathIR task[1], and a queryset containing 19 Wikipedia Main Task queries and 4 Wikipedia Browsing Task queries. Well known trec_eval[2] tool is employed to evaluate the system's effectiveness on the grounds of Precision at 5 (*P_5*), Precision at 10 (*P_10*), mean average precision (*map*) and binary preference-based measure (*bpref*) measures. The tool compares system results against a Gold Dataset containing judged entries for each query of the queryset.

Considerably high values of the evaluation measures serve as testament to the distinguished abilities of the proposed approach. Furthermore, the system results are also compared with the search results produced by a conventional text-based search engine. Vast gaps in the corresponding evaluation measures of the two systems are indicative of the conceptual difference between the conventional text-based search and the math formulae search.

Rest of the paper is structured as follows: Section 2 reviews some closely related works; Section 3 describes system architecture and its working description; Section 4 discusses experimental framework used for testing effectiveness of the system; Section 5 presents system results and their comprehensive analysis; Section 6 points direction for future research; Section 7 concludes the paper.

[1] http://ntcir-math.nii.ac.jp/

[2] http://trec.nist.gov/trec_eval/

2 Related Works

Information rich contents of the scientific documents serve as crucial input to many scientific and technical research. However, scientific documents being primarily rich in math formulae, conventional text-based indexing and search techniques often fail to retrieve information from such documents.

Although a number of past works have addressed the issue of information retrieval from scientific documents, the formulae embedding approach, in particular, is still in its infancy. A relevant work in this regard is the document retrieval system [17] of NTCIR-12 MathIR task, which uses Document To Vector (Doc2Vec) and Latent Dirichlet Allocation (LDA) for retrieving similar formulae and sub-formulae. More specifically, the Distributed Bag of Words (PV-DBOW) model, a special variant of the Doc2Vec algorithm, is exploited and extended to represent math expressions in terms of the real valued vectors.

Furthermore, the preliminary exploration of formulae embedding has shown promising results in context of math information retrieval [4]. Initially a symbol2vec method transforms formulae symbols into vector representation, and the cosine distance of symbol vectors of closely related symbols turn out to be minimum. Symbol vectors and Distributed Memory Model of Paragraph Vectors (PV-DM) are then used to embed formulae. Well known cosine similarity measure computes similarity of formulae vectors, and suitable scores are assigned to the retrieved search results. However, the insufficient system description and analysis of results barely convey strengths and weaknesses of the approach.

The necessity of math formulae search led to the introduction of a new math pilot task in NTCIR-10 conference [1, 6], and the task has continued to be part of subsequent NTCIR conferences, namely NTCIR-11 [2, 5] and NTCIR-12 [18, 7]. The Math Indexer and Searcher (MIaS) system of NTCIR-10 conference employs preprocessing of scientific documents followed by canonicalization and linearization of mathematical expressions to ease their retrieval [8]. An enhanced version of MIaS [13] uses better preprocessing, canonicalization, math representation and query expansion strategies. Furthermore, combining text keywords with math query using different querying strategies has furthered the effectiveness of retrieval [9]. MIaS at NTCIR-12 conference was characterized by an additional structural unification component, which helped retrieve semantically similar formulae [14]. Tangent-3 math retrieval system [3] at NTCIR-12 uses inverted indexing to store mathematical entities extracted from Symbol Layout Tree (SLT). This is followed by a 2 stage retrieval process. While the first stage primarily concerns retrieval of relevant expressions using iterator trees and trivial ranking, the second stage concerns strict re-ranking of top-k best candidates.

Modifications in conventional indexing and search techniques have also contributed notably to the domain of math formulae search. In particular, substitution tree based indexing techniques [15, 12] index math formulae along nodes and leaves of a substitution tree and, hence, minimize the memory requirement. Nodes of the tree correspond to substitutions whereas the leaves correspond to mathematical entities. Depth first traversal of the substitution tree yields an indexed formula. An improved and intelligent boolean model [11] modifies conventional AND search mechanism, and performs ORing and stemming of the user query for effective retrieval of the scientific documents.

An architecture for scientific document retrieval, containing 3 different Text-Text, Text-Math and Math-Math entailment modules, is proposed in [10]. Architecture supports search for user query containing text as well as mathematical contents. Text Entailment (TE) module matches text part of the query to the indexed text contents, Math Entailment (ME) module matches mathematical part of query to the indexed math formulae and Text Math Entailment (TME) module matches text part of the query to standard names of the indexed math formulae.

3 System Description

This section provides details of the corpus used for experimentation, and the crucial components of the system architecture which collaboratively function to output relevant search results for each user

query. Figure 1 shows the system architecture and workflow of the proposed system.

3.1 Corpus Description

Proposed system has been experimented with a corpus containing 212 documents chosen from vast arXiv and Wikipedia corpora of NTCIR-12 MathIR task. Total size of the corpus is 22.6 MB, with majority of the documents being formulae rich and considerably large in size. The documents are in XHTML format, with formulae encoded using MathML.

A) **arXiv corpus**
The arXiv corpus of NTCIR-12 MathIR task comprises of 105,120 scientific documents chosen from the following arXiv categories: math, cs, physics:math-ph, stat, physics:hep-th and physics:nlin [18]. Documents contain roughly 60 million math formulae written using MathML. arXiv corpus is intended for technical users.

B) **Wikipedia corpus**
The Wikipedia corpus of NTCIR-12 MathIR task comprises of 319,689 articles in XHTML format, which contain 590,000 formulae encoded using LaTeX, Presentation MathML and Content MathML. Unlike arXiv corpus, Wikipedia corpus is intended for non-technical users.

While selecting the documents for experimentation, adequate care has been taken to ensure that:

1. a reasonable proportion of the documents are common to different queries of the query set.
2. some documents contain exact formula query, whereas some contain similar formulae, sub-formulae and parent formulae.

Table 1 summarizes the corpus information.

3.2 Document Preprocessor

Core focus of the proposed approach being formulae retrieval and formulae matching, the document preprocessor extracts only MathML formulae from the documents containing text as well as math contents. Figure 2 shows a sample XHTML document containing text as well as math contents. Figure 3(a) shows the output of document preprocessor, wherein the text contents have been removed.

3.3 Formula Preprocessor

Primary job of the formula preprocessor is to filter out unnecessary MathML elements and attributes, which barely contribute to the information content of the formula. Table 2 shows some examples of preprocessing done by the formula preprocessor. Also, Figure 3(b) shows a complete output of the formula preprocessor, wherein only relevant information of the raw formula have been preserved.

3.4 Formula Embedding Module

The formula embedding module takes processed formula as input and outputs a corresponding binary vector. Careful observation of the Presentation MathML formulae guides us to the fact that ⟨*mi*⟩ and ⟨*mo*⟩ elements form the key constituents of nearly all formulae. Thus, the module parses processed MathML formula to fetch contents from all occurrences of ⟨*mi*⟩ and ⟨*mo*⟩ elements. Besides, the module also accounts for all other MathML elements, such as ⟨*msqrt*⟩, ⟨*msub*⟩, ⟨*msup*⟩ and so on, present in the formula. Eventually, the module uses fetched information contents and the bit position information table to set respective bits of the corresponding formula vector, which has a fixed length of 150 bits.

A bit position information table (see Table 3) depicts bit positions assigned to different mathematical entities. A vector of length 150 suffices to encode mathematical contents of all the documents in our corpus. However, the module offers flexibility to vary (increase or decrease) length of formula vector, if required.

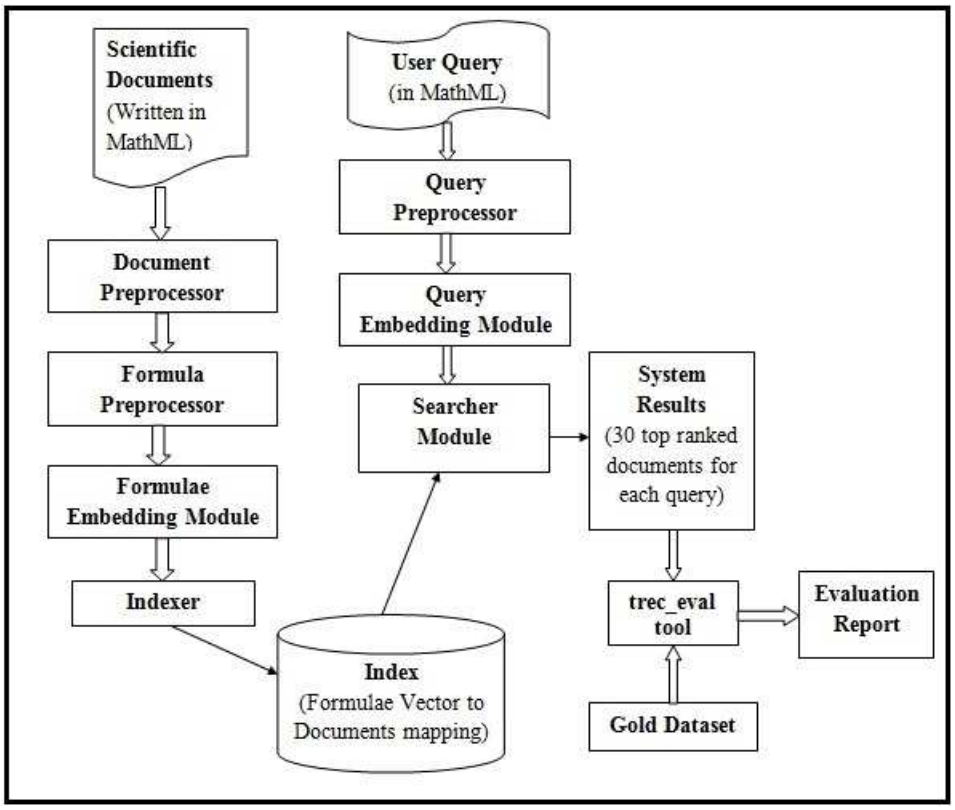

Fig. 1. System architecture and working description

Table 1. Corpus Description

Size of Corpus	Number of Documents	Source of Documents	Format of Documents	Encoding of Formulae
22.6 MB	212	arXiv & Wikipedia corpora of NTCIR-12 MathIR task	XHTML	Presentation & Content MathML

Figure 4 shows typeset representation and formula vector of a processed MathML formula.

The following points are worth noting about bit position information table and formula vector:

1. Bit positions 0–25, 57–65 and 71–100 correspond to the content of ⟨*mi*⟩ tag, bit positions 26–45, 66–70 and 101–149 refer to the contents of ⟨*mo*⟩ tag, and bit positions 46–56 refer to essential MathML tags which contribute to the semantics of formula.
2. The list of entities is by no means exhaustive, but it accounts for nearly all the entities contained in our corpus. In future version of the proposed system, the length of formula vector will be extended to account for more math symbols and their semantics.

```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="application/xhtml+xml; charset=UTF-8" />
</head><body>
<div class="ltx_para" id="p7">
<p class="ltx_p" id="p7.1">In other words, while any classical Lagrangian and any
wave equation (or field equation in the case of second
quantization) is an evolution of Newton's second law
<math xmlns="http://www.w3.org/1998/Math/MathML" xref="p7.1.m1.1.cmml" alttext=
"{\bf F}=m{\bf a}" class="ltx_Math" id="p7.1.m1.1" display="inline"><semantics
xref="p7.1.m1.1.cmml" id="p7.1.m1.1a"><mrow xref="p7.1.m1.1.5.cmml" id="p7.1.m1.1.5">
<mi xref="p7.1.m1.1.1.cmml" id="p7.1.m1.1.1">F</mi><mo xref="p7.1.m1.1.2.cmml"
 id="p7.1.m1.1.2">=</mo><mrow xref="p7.1.m1.1.5.1.cmml" id="p7.1.m1.1.5.1">
<mi xref="p7.1.m1.1.3.cmml" id="p7.1.m1.1.3">m</mi><mo xref="p7.1.m1.1.5.1.1.cmml"
 id="p7.1.m1.1.5.1.1"></mo><mi xref="p7.1.m1.1.4.cmml" id="p7.1.m1.1.4">a</mi></mrow>
</mrow><annotation-xml xref="p7.1.m1.1" id="p7.1.m1.1.cmml" encoding="MathML-Content">
<apply xref="p7.1.m1.1.5" id="p7.1.m1.1.5.cmml">
<eq xref="p7.1.m1.1.2" id="p7.1.m1.1.2.cmml"></eq><ci xref="p7.1.m1.1.1"
id="p7.1.m1.1.1.cmml">F</ci><apply xref="p7.1.m1.1.5.1" id="p7.1.m1.1.5.1.cmml">
<times xref="p7.1.m1.1.5.1.1" id="p7.1.m1.1.5.1.1.cmml"></times><ci xref="p7.1.m1.1.3"
 id="p7.1.m1.1.3.cmml">m</ci><ci xref="p7.1.m1.1.4" id="p7.1.m1.1.4.cmml">a</ci>
</apply></apply></annotation-xml><annotation xref="p7.1.m1.1.cmml" id="p7.1.m1.1b"
 encoding="application/x-tex">{\bf F}=m{\bf a}</annotation></semantics></math>,
In the approach mentioned above
inertia follows from the anisotropic radiation pressure. In
this way, inertia appears as a consequence of the third law,
i.e., of momentum conservation, and ultimately of
translation invariance.</p>
</div>
</body></html>
```

Fig. 2. A sample XHTML document containing text as well as math formulae

```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="application/xhtml+xml; charset=UTF-8"/>
</head><body><math xmlns="http://www.w3.org/1998/Math/MathML" xref="p7.1.m1.1.cmml"
 alttext="{\bf F}=m{\bf a}" class="ltx_Math" id="p7.1.m1.1" display="inline">
<semantics xref="p7.1.m1.1.cmml" id="p7.1.m1.1a"><mrow xref="p7.1.m1.1.5.cmml"
id="p7.1.m1.1.5"><mi xref="p7.1.m1.1.1.cmml" id="p7.1.m1.1.1">F</mi><mo xref=
"p7.1.m1.1.2.cmml" id="p7.1.m1.1.2">=</mo><mrow xref="p7.1.m1.1.5.1.cmml"
id="p7.1.m1.1.5.1"><mi xref="p7.1.m1.1.3.cmml" id="p7.1.m1.1.3">m</mi>
<mo xref="p7.1.m1.1.5.1.1.cmml" id="p7.1.m1.1.5.1.1"></mo><mi xref="p7.1.m1.1.4.cmml"
 id="p7.1.m1.1.4">a</mi></mrow></mrow><annotation-xml xref="p7.1.m1.1"
id="p7.1.m1.1.cmml" encoding="MathML-Content"><apply xref="p7.1.m1.1.5"
 id="p7.1.m1.1.5.cmml"><eq xref="p7.1.m1.1.2" id="p7.1.m1.1.2.cmml"></eq>
<ci xref="p7.1.m1.1.1" id="p7.1.m1.1.1.cmml">F</ci><apply xref="p7.1.m1.1.5.1"
id="p7.1.m1.1.5.1.cmml"><times xref="p7.1.m1.1.5.1.1" id="p7.1.m1.1.5.1.1.cmml">
</times><ci xref="p7.1.m1.1.3" id="p7.1.m1.1.3.cmml">m</ci><ci xref="p7.1.m1.1.4"
id="p7.1.m1.1.4.cmml">a</ci></apply></apply></annotation-xml><annotation
xref="p7.1.m1.1.cmml" id="p7.1.m1.1b" encoding="application/x-tex">{\bf F}
=m{\bf a}</annotation></semantics></math>
</body>
</html>
```

(a)

```
<?xml version="1.0" encoding="utf-8"?><html xmlns="http://www.w3.org/1999/xhtml">
<head><meta http-equiv="Content-Type" content="application/xhtml+xml; charset=UTF-8"/>
</head><body><math><semantics><mrow><mi>F</mi><mo>=</mo><mrow><mi>m</mi><mo></mo><mi>
a</mi></mrow></mrow> </semantics></math></body></html>
```

(b)

Fig. 3. (a) Output of Document Preprocessor (b) Output of Formula Preprocessor

3. In order to retrieve specific results ahead of non-specific ones, distinction is maintained between single alphabet variables, by assigning them different bit positions ranging from 0–25. However, the table makes no distinction between the cases of variables,

Table 2. Preprocessing done by Formula Preprocessor

Original	Processed
⟨*mi xref*=*"p7.1.m1.1.1.cmml" id*=*"p7.1.m1.1.1"*⟩	⟨*mi*⟩
⟨*mo xref*=*"p7.1.m1.1.2.cmml" id*=*"p7.1.m1.1.2"*⟩	⟨*mo*⟩
⟨*math xmlns*=*"http://www.w3.org/1998/Math/MathML"*⟩	⟨*math*⟩
⟨*semantics xref*=*"p7.1.m1.1.1.cmml" id*=*"p7.1.m1.1.1"*⟩	⟨*semantics*⟩
⟨*annotation-xml*⟩...⟨/*annotation-xml*⟩	Contents between the two elements are removed
⟨*annotation*⟩...⟨/*annotation*⟩	Contents between the two elements are removed

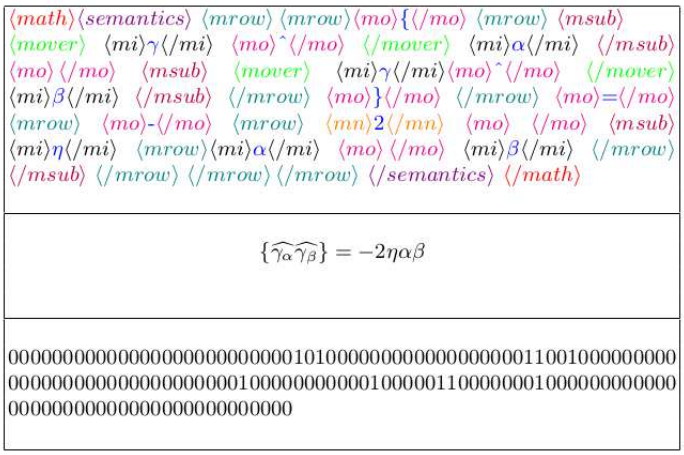

Fig. 4. Typeset Representation and Formula Vector of a processed MathML formula

and the different cases of same variable are assigned same bit position. For example, 'a' and 'A' are assigned same bit position equal to 0. Furthermore, since the entities "exp" and 'e' are interchangeably used, they have been assigned same bit position equal to 4. For the same reason, "$\log$" and "$\ln$" are assigned same bit position.

4. Proposed system seeks generalized results for the query term involving trigonometric ratios, such as *"sin"*, *"cos"*, *"tan"*, *"cot"*, and so on. Thus, all such ratios are assigned same bit position equal to 90.

5. An entity which can be part of ⟨*mi*⟩ as well as ⟨*mo*⟩ tags, i.e. the one which can act as variable as well as operator, is assigned two distinct bit positions. One such entity is Σ, which is assigned bit positions 76 and 112, designating variable and operator, respectively.

6. Bit position 65 designates a multi character variable, whose name is not a standard variable name (such as, lim, log, gcd, and so on).

7. The formula vector does not account for multiple occurrences of the entities. Even though an entity occurs more than once in the formula, only one corresponding bit of the

formula vector is set. This limitation, however, causes inability to retrieve relevant search results, if the user query contains repetition of an entity.

However, none of the above constraints incurs loss of generality, and the approach is scalable.

3.5 Indexer

Indexer stores formula vector to corresponding document mapping in an index. System indexes a total of 5,969 processed formulae, derived from 212 documents of the corpus.

Moreover, the index size is 1 MB, which is 4.42% of the corpus size. Corpus size and the size of formula vector are the primary factors, which affect size of the index. Indexer indexes formulae in document-wise fashion, meaning thereby that only after having indexed all the formulae of a document, it goes for indexing next document of the corpus.

However, it is observed that '0' bits of a formula vector, which designate absence of mathematical entities, unnecessarily increase the size of index. Our primary concern being presence of an entity, future modification will refrain from storing '0' bit information in the index.

3.6 Query Preprocessor

The task of query preprocessor resembles that of document and formula preprocessors. It extracts math formula from the user query (text+math) and processes it to remove unnecessary MathML elements and attributes.

3.7 Query Embedding Module

Similar to the formula embedding module, the query embedding module transforms processed user query into a binary query vector. An important observation is that some NTCIR queries comprise of query variable (qvar), which may designate a variable name, numeric value or complex expression. As of now, the module assumes *"qvar"* to be numeric value and sets the bit position 47, corresponding to element $\langle mn \rangle$, equal to 1 if *"qvar"* is encountered.

3.8 Searcher Module

Given a user query in vector form, primary objective of the searcher module is to retrieve relevant search results. Module uses number of matching set bits of query vector and formula vector as the criterion to judge relevance of formula vector and, hence, the search results. Although the criterion is not foolproof, it works effectively well for majority of user queries. Probable errors and associated problems of the selected criterion are comprehensively analyzed, with suitable examples, in subsection 5.4.

The module retrieves 30 most similar top ranked documents corresponding to each user query. It should be noted that a query vector may match different formulae of the same document with different scores, and if such a document succeeds to find place in the result list, the module reports highest of all the scores. In any case, the module refrains from reporting redundant results, which may hamper the evaluation process.

Eventually, the system results are fed to *trec_eval* evaluation tool, which compares them with the Gold dataset entries and outputs an evaluation report, which summarizes effectiveness of the system in terms of a number of parameters.

4 Experimental Design

This section details queryset, gold dataset and evaluation parameters used to evaluate performance of the system.

4.1 Queryset Description

Our queryset comprises of 23 MathML queries, derived from NTCIR-12 MathIR Wikipedia Main Task and Wikipedia Formula Browsing Task querysets. The queryset is a mix of simple and complex queries, and each query has an associated QueryID. While selecting documents for the corpus, it is ensured that the documents contain either exact form or similar form or sub-formula form of the query formulae in the queryset. Each of the queries in the queryset is transformed into a query vector by the query embedding module.

Table 3. Bit Position Information Table

Entity	Position	Entity	Position	Entity	Position	Entity	Position
a/A to z/Z	0-25	⟨msqrt⟩	56	log, ln	88	±	120
exp	4	$\mathbb{Z}$	57	!	89	$\int$	121
=	26	$\mathbb{N}$	58	Trigo. Ratios	90	∘	122
Product	27	$\mathbb{Q}$	59	$\mathbb{R}$	91	′	123
-	28	¬	60	θ	92	∧	124
,	29	α	61	gcd	93	∃	125
+	30	γ	62	xor	94	¬	126
∇	31	ω	63	τ	95	lim	127
∂	32	ϑ	64	η	96	<,⟨	128
→	33	Var. Name	65	σ	97	>, ⟩	129
.	34	Null	66	Ω	98	⊗	130
(	35	†	67	#	99	⊢	131
)	36	:	68	⌜	100	⊓	132
≡	37	dist	69	≠	101	⊔	133
≫	38	∓	70	{	102	∥	134
∝	39	ϕ, φ, Φ	71	}	103	∪	135
≈	40	$\hbar$	72	⊙	104	∩	136
/	41	π	73	≤	105	↦	137
⊆	42	△	74	∈	106	⊂	138
⊕	43	μ	75	[	107	det	139
∼	44	Σ	76	]	108	∏	140
|	45	∈	77	∗, x	109	mod	141
⟨mfrac⟩	46	...	78	∉	110	sup	142
⟨mn⟩	47	δ	79	ˆ	111	≥,⩾,≳	143
⟨msub⟩	48	ψ, Ψ	80	Σ	112	dim	144
⟨msup⟩	49	Γ	81	;	113	:=	145
⟨msubsup⟩	50	∞	82	¯	114	≅	146
⟨mover⟩	51	ρ	83	⟺ , ⇔	115	max	147
⟨munderover⟩	52	β	84	⇒	116	inf	148
⟨munder⟩	53	λ	85	⌈	117	min	149
⟨mtable⟩	54	ξ	86	⌉	118		
⟨mmultiscripts⟩	55	□	87	∀	119		

4.2 Gold Dataset

Gold Dataset (also known as qrel file) strictly adheres to the Text REtrieval Conference (TREC) qrel format[3], and contains a set of human assessed documents for each query in the queryset. Gold dataset has the format:

QueryID	*Iteration*	*Document#*	*Relevance*

where, *QueryID* designates specific query, *Iteration* is an irrelevant field usually set to 0 and ignored by the trec_eval tool, *Document#* specifies document number of the retrieved document and *Relevance* designates binary judgment (1 for relevant and 0 for irrelevant). A document is judged relevant even if it contains small content of the query formula. Our Gold Dataset contains 690 entries, wherein 77 entries are judged irrelevant. Figure 5 shows snapshot of the Gold Dataset.

[3] http://trec.nist.gov/data/qrels_eng/

QueryID	Iteration	Document#	Relevance
18	0	math-ph0203052_1_31	1
18	0	hep-th0109074_1_16	0
18	0	math-ph0301038_1_94	1
18	0	math0111030_1_52	0
18	0	math0103171_1_74	1
18	0	math0009189_1_16	1
18	0	math0009238_1_21	0
18	0	math0009184_1_22	1
18	0	math-ph0203052_1_28	1
18	0	math0703396_1_15	1
18	0	math0604510_1_37	1
18	0	hep-th0209066_1_12	0
18	0	math0105098_1_34	0
18	0	math0104234_1_56	1
18	0	hep-th0210069_1_4	0
18	0	nlin0207022_1_19	0
19	0	nlin0402026_1_78	1
19	0	1201.0676_1_114 1	
19	0	math0402085_1_108	1
19	0	1202.3313_1_59 1	
19	0	math0009180_1_31	1
19	0	math0009212_1_60	1
19	0	math0009244_1_111	1
19	0	math0101072_1_9 1	
19	0	math-ph0203052_1_31	1
19	0	math0002036_1_233	1

Fig. 5. Snapshot of Gold Dataset

4.3 Evaluation Parameters

System results are evaluated on the grounds of *P_5*, *P_10*, *map* and *bpref*. All these measures are computed for each query, and individual measures are then averaged over all the queries in the queryset. *P_k*, with *k* being equal to 5 and 10, refers to the count of relevant documents out of first *k* retrieved documents. In order to compute *map*, precision score is computed whenever a relevant document is retrieved.

The precision scores are then averaged over all the relevant documents, corresponding to a query, to get average precision. Average precision scores are averaged over all the queries to get *map*. Furthermore, *bpref* measures ability of the system to retrieve relevant documents ahead of irrelevant ones. Besides, we have also computed fraction of the relevant documents which are retrieved (denoted as *frac_ret*). Count of retrieved relevant documents (*num_rel_ret*) divided by the count of relevant documents (*num_rel*) gives the measure of *frac_ret*. Values of *num_rel* and *num_rel_ret*, used to compute *frac_ret*, are obtained from the evaluation report. All the five parameters range from 0 to 1, with larger signifying the better.

5 Results and Analysis

5.1 Format of Result Set

The Result Set containing system results adheres to the result file format of TREC. Each tuple of the result set is of the form:

QueryID Iteration Document# Rank Similarity Run_ID

Although *Iteration (iter)* and *Rank* fields are mandatory, the two are ignored by the evaluation tool. *Similarity (sim)* is usually float in nature and attains larger value for the documents which are retrieved first. In our case, the count of matching set bits refers to the *Similarity* score. *Run_ID*, a trivial field, is a string which characterizes system run and gets printed on the evaluation report. The three trivial fields, namely *Iteration*, *Rank* and

Run_ID, have been set to dummy values *"Q0"*, *"20"* and *"demo"*, respectively. Figure 6 shows snapshot of the Result Set.

5.2 Evaluation Report

Evaluation report generated by the trec_eval contains values of a number of parameters, which summarize effectiveness of the proposed system. However, only 5 parameters, discussed in the subsection 4.3, are used to infer the system's effectiveness. Labeled bar chart, shown in Figure 7, depicts values of the evaluation parameters for the proposed system. Fairly high values of the parameters substantiate effectiveness of the proposed approach. Out of a total of 613 relevant documents, system succeeds in retrieving 535 documents, which equates to 87.27% (frac_ret=0.8727) of the total relevant documents.

5.3 Comparison with Conventional Text Search Engine

The same experimental framework (Corpus, Gold Dataset and Query Set) is used to retrieve results from Apache Nutch[4] based conventional text search engine. Labeled bar chart, shown in Figure 8, depicts comparison of evaluation parameters for the two systems. Significant differences in corresponding evaluation measures are attributed to the fact that math search is way different than conventional text search in terms of retrieval needs of the end users, and the way query term is matched to the indexed terms. In particular, the incompetence of conventional text search engine is reflected in its ability to retrieve only 54 out of 613 relevant documents, which equates to 8.81% (frac_ret=0.0881) of the total relevant documents.

5.4 Results Analysis

In this subsection, we have comprehensively analyzed strengths and possible limitations of proposed method from different perspectives. Following observations and system outcomes are worth considering:

[4] http://nutch.apache.org/

1. First observation is that given a query formula, system flawlessly retrieves all search results, wherein the query term is either present in its exact form or present as sub-formula of a larger parent formula. This actually happens because similarity score (number of matching set bits of query vector and the search result) for such results will be maximum. For example, Table 4 shows some of the search results for the user query $a \oplus b$. In first and second search results, the query term appears in its exact form whereas in third and fourth search results, it appears as sub-formula of its parent formula. Moreover, the maximum similarity score is 3 as the query formula comprises of three distinct entities, namely a, $\oplus$ and b.

2. Second observation is that even the complex formulae, which share semantic relatedness with the user query and contain all its entities, are successfully retrieved. For example, the fifth and sixth search results (see Table 4) neither contain exact query formula nor its parent form, but they are retrieved by our system, as they semantically resemble the query formula and contain all its entities.

3. Third observation is that the proposed system successfully retrieves sub-formulae and similar formulae of a given user query. Consider, for example, the search results for user query $O(mn \log m)$, shown in Table 5.

   Snippets shown in first four search results depict sub-formulae of the user query. Also, the fifth search result is semantically similar to the user query.

4. Fourth observation is that although the decision of correlating relevance with high similarity score predominantly works well, it does incur failure in certain cases. Consider, for example, the search results for the user query ${}_2F_1(a, b; c; z)$, shown in Table 6.

   Though the first three search results are appreciable, the last search result is absolutely irrelevant. Such an irrelevant result succeeds in finding place in the result list, because it

qid	iter	Document#	Rank	sim	Run_ID
18	Q0	hep-th0210024_1_23	20	8	demo
18	Q0	math-ph0203052_1_31	20	8	demo
18	Q0	hep-th0109074_1_16	20	8	demo
18	Q0	math-ph0301038_1_94	20	8	demo
18	Q0	math0111030_1_52	20	8	demo
18	Q0	math0103171_1_74	20	8	demo
18	Q0	math0009189_1_16	20	8	demo
18	Q0	math0009238_1_21	20	8	demo
18	Q0	math0009184_1_22	20	8	demo
18	Q0	math-ph0203052_1_28	20	8	demo
18	Q0	math0703396_1_15	20	8	demo
18	Q0	math0604510_1_37	20	8	demo
18	Q0	hep-th0209066_1_12	20	8	demo
19	Q0	math-ph0203052_1_31	20	15	demo
19	Q0	math0002036_1_233	20	14	demo
19	Q0	quant-ph0212072_1_20	20	13	demo
19	Q0	0812.4450_1_28	20	13	demo
19	Q0	math0104234_1_56	20	13	demo
19	Q0	hep-th0210069_1_4	20	13	demo
19	Q0	math0205147_1_31	20	13	demo
19	Q0	math0009244_1_111	20	13	demo
19	Q0	math0604510_1_37	20	13	demo

Fig. 6. Snapshot of Result Set

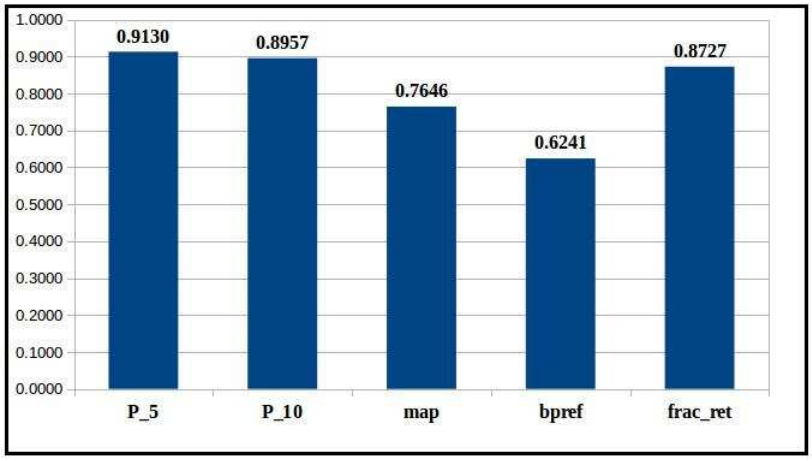

Fig. 7. Bar graph showing values of evaluation parameters for the proposed system

contains majority of the query formula entities, namely *'A'*, *'B'*, *'C'*, *'('*, *')'* and *'1'*.

5. Fifth observation is that for the user query $F = ma$, system retrieves an irrelevant search result ($\int_a^m F(x)dx$), containing all the entities of the query formula, among top-ranked search results. More importantly, the system fails to retrieve a relevant search result ($F = bc$) primarily because of its low similarity score equal to 2, and the constraint laid on the system to retrieve only top 30 documents. Although an increase in the search result

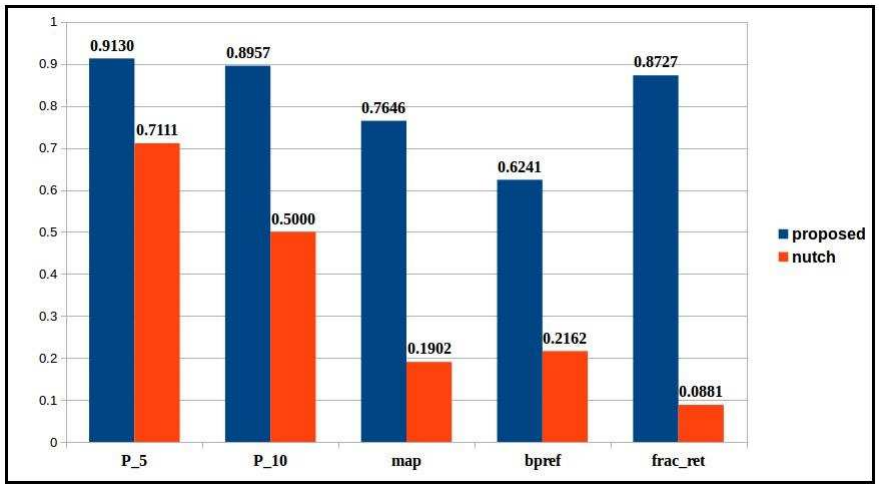

Fig. 8. Bar graph showing comparison of proposed system and conventional text search engine

Table 4. Search results for the user query $a \oplus b$

Search Results	Documents	Similarity Score
$a \oplus b$	hep-th0005087_1_93	3
$b \oplus a$	1310.4652_1_21	3
$a \oplus b = b$	1209.1718_1_16	3
$(a, b) \in A \oplus B$	math0011063_1_119	3
$M(B) = M(A) \oplus M(A^{-1}B)$	math0102078_1_5	3
${W_P}^k = {A_P}^k \oplus {B_P}^k$	math0206213_1_24	3

Table 5. Search results for the user query $O(mn \log m)$

Search Results	Documents	Similarity Score
$O(\log n$ +k $)$	0805.1348_1_4	5
$O(\sqrt{\log n})$	quant-ph0205083_1_82	5
$O(n^{\frac{1}{\log_p m}})$	quant-ph0211179_1_57	5
$O(\log n)$	0805.1348_1_32	5
$O(nt \log n)$	1308.3898_1_18	5

Table 6. Search results for the user query ${}_2F_1(a, b; c; z)$

Search Results	Documents	Similarity Score
${}_2F_1(a + 1, 1; 2; z)$	math-ph0203052_1_8	10
${}_2F_1(k + 1, k + 1; 1; \lambda)$	quant-ph0212072_1_20	7
${}_2F_1(-n, b; \gamma; y)$	math-ph0203052_1_21	7
$\widetilde{h} = \widetilde{W}^{-1} dx^2 + h_{AB}(x, x^C) dx^A dx^B$	hep-th0108170_1_12	7

limit will undoubtedly retrieve such results, the intelligence lies in the system's ability to retrieve such formulae, among top-ranked documents, under the imposed constraint. Moreover, the system should also invalidate irrelevant search results, which contain query formula entities but posses little or no semantic relatedness with the query.

Our fourth and fifth observations guide us to the fact that using number of matching set bits of the formula vector and the query vector as similarity measure criterion is not foolproof. A search result with low similarity score might prove to be more relevant than the one with better score. Thus, even though the proposed system lives up to our expectations as a baseline system, a future extension necessitates revision in similarity measure criterion to overpower existing inabilities.

6 Future Directions

As the system is baseline and first of its kind, the list of possible future directions goes long. Some salient future directions are undermentioned:

1. Proposed system currently refrains from indexing and searching text contents. Thus, augmenting baseline system with text indexing and searching ability can further the system's performance, and possibly invalidate some of the irrelevant search results.
2. Formula and Query vectors need to be lengthened to accommodate new symbols and semantic information. For example, some bit positions may account for count of occurrences of an entity or co-occurrence of two or more entities, which will eventually let the searcher module prefer relevant results over irrelevant ones.
3. Similarity measure criterion can be modified to assign weightage to certain bit positions. It is a well-established fact that the end users may compromise the variable names but not the operators. For example, given user query to be $a \oplus b$, the end user will be more content with $c \oplus d$ than $a + b$, even though the similarity score for the latter will be more than the former. Thus, assigning more weightage to operator bit positions, while computing similarity score, can do wonders.
4. It can be fascinating to prompt the end users for key entities in their queries. Search results not containing such entities should not be retrieved. This can be ensured by dynamically assigning more weightage to the bit positions of key entities while computing the similarity score.

7 Conclusion

In this paper, we describe an unconventional formula embedding approach, which can ease retrieval of math formulae present inside scientific documents. Uncommonly used formula embedding approach opens door to the new horizons and helps combat underlying challenges of the scientific document retrieval. Having transformed the processed document formulae and the user query into vectors of fixed size, the proposed system uses number of matching set bits of the document formula and the query vector as similarity measure to retrieve and rank the indexed formulae. Comprehensive analysis of the system results affirms underlying strengths and weaknesses of the system. Although the approach has associated limitations, it shows competence in retrieving exact query formula, parent formulae, sub-formulae and similar formulae. Modifying similarity measure criterion, and incorporating support for indexing and searching of textual terms are some of the notable future modifications, which can further the effectiveness of retrieval.

Acknowledgement

The work presented here falls under the Research Project Grant No. YSS/2015/000988 and supported by the Department of Science & Technology (DST) and Science and Engineering Research Board (SERB), Govt. of India. The authors would like to acknowledge the Department of Computer Science & Engineering, National

Institute of Technology Mizoram, India for providing infrastructural facilities and support.

References

1. **Aizawa, A., Kohlhase, M., & Ounis, I. (2013).** Ntcir-10 math pilot task overview. *Proceedings of the 10th NTCIR Conference*, Tokyo, Japan, pp. 654–661.
2. **Aizawa, A., Kohlhase, M., Ounis, I., & Schubotz, M. (2014).** Ntcir-11 math-2 task overview. *Proceedings of the 11th NTCIR Conference*, Tokyo, Japan, pp. 88–98.
3. **Davila, K., Zanibbi, R., Kane, A., & Tompa, F. W. (2016).** Tangent-3 at the ntcir-12 mathir task. *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*, Tokyo, Japan, pp. 338–345.
4. **Gao, L., Jiang, Z., Yin, Y., Yuan, K., Yan, Z., & Tang, Z. (2017).** Preliminary exploration of formula embedding for mathematical information retrieval: Can mathematical formulae be embedded like a natural language? *arXiv preprint arXiv:1707.05154*.
5. **Joho, H. & Kishida, K. (2014).** Overview of ntcir-11. *Proceedings of the 11th NTCIR Conference*, Tokyo, Japan, pp. 9–12.
6. **Joho, H. & Sakai, T. (2014).** Overview of ntcir-10. *Proceedings of the 10th NTCIR Conference*, Tokyo, Japan, pp. 1–7.
7. **Kishida, K. & Kato, M. P. (2016).** Overview of ntcir-12. *Proceedings of the 12th NTCIR Conference*, Tokyo, Japan, pp. 1–7.
8. **Líška, M., Sojka, P., & Ružicka, M. (2013).** Similarity search for mathematics: Masaryk university team at the ntcir-10 math task. *Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies*, Tokyo, Japan, pp. 686–691.
9. **Líška, M., Sojka, P., & Ružicka, M. (2015).** Combining text and formula queries in math information retrieval: Evaluation of query results merging strategies. *Proceedings of the First International Workshop on Novel Web Search Interfaces and Systems*, ACM, Melbourne, Australia, pp. 7–9.
10. **Pakray, P. & Sojka, P. (2014).** An architecture for scientific document retrieval using textual and math entailment modules. *Proceedings of Recent Advances in Slavonic Natural Language Processing*, Karlova StudÃ¡nka, Czech Republic, pp. 107–117.
11. **Pathak, A. & Pakray, P. (2018).** An improved and intelligent boolean model for scientific text information retrieval. *Communications in Computer and Information Science (CCIS)*, volume 836, Springer, pp. 465–476.
12. **Pathak, A., Pakray, P., Sarkar, S., Das, D., & Gelbukh, A. (2017).** Mathirs: Retrieval system for scientific documents. *Computación y Sistemas*, Vol. 21, No. 2, pp. 253–265.
13. **Ružicka, M., Sojka, P., & Líška, M. (2014).** Math indexer and searcher under the hood: History and development of a winning strategy. *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies*, Tokyo, Japan, pp. 127–134.
14. **Ruzicka, M., Sojka, P., & Líska, M. (2016).** Math indexer and searcher under the hood: Fine-tuning query expansion and unification strategies. *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*, Tokyo, Japan, pp. 331–337.
15. **Schellenberg, T., Yuan, B., & Zanibbi, R. (2012).** Layout-based substitution tree indexing and retrieval for mathematical expressions. *Proceedings of the Document Recognition and Retrieval*, California, USA, pp. 1–8.
16. **Sojka, P. & Líška, M. (2011).** The art of mathematics retrieval. *Proceedings of the 11th ACM symposium on Document engineering*, California, USA, pp. 57–60.
17. **Thanda, A., Agarwal, A., Singla, K., Prakash, A., & Gupta, A. (2016).** A document retrieval system for math queries. *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*, Tokyo, Japan, pp. 346–353.
18. **Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topic, G., & Davila, K. (2016).** Ntcir-12 mathir task overview. *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*, Tokyo, Japan, pp. 299–308.

Article received on 25/01/2018; accepted on 05/03/2018.
Corresponding author is Amarnath Pathak.